

How Emotet Infects a PC

In This Document, we are going to discuss **How Emotet Infects** a PC and a detailed analysis of it.

What is Emotet?

Emotet is a banking malware, I think you have heard this quote before, but from where the “banking” term came?

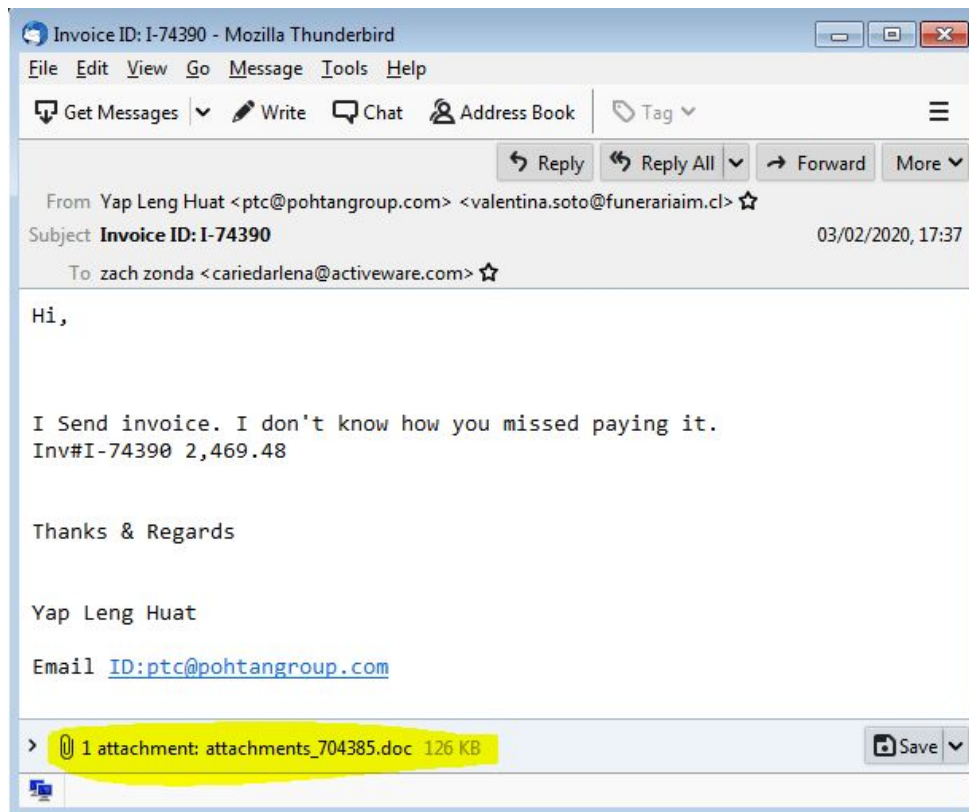
This malware sneaks into your PC and steals your personal information like username, passwords, in the very beginning of its era it is focused on stealing internet banking related information.

Emotet was first encountered in 2014. As it got aged it evolved from just a **banking** trojan to perform many more things.

Now it has the ability to infect the host to send spam email, self upgrading modules, backdoor activity, etc..

How does it spread?

It usually got spread by the spam emails, just like in the screenshot below.



Would you open this invoice, if it comes to your inbox? Maybe you don't, but there are still a huge number of people who are not aware of malicious emails and they download the attachment and open it and that's enough to infect your PC.

What is in the Attachment?

Attachment is just a doc file. Do you think it can do any harm to your PC?
Let's figure out what it can do

> 1 attachment: attachments_704385.doc 126 KB

SHA-1 of the doc file: 0dc994ee4c96f4fcd8f8d4a5b3fadba14f15355e

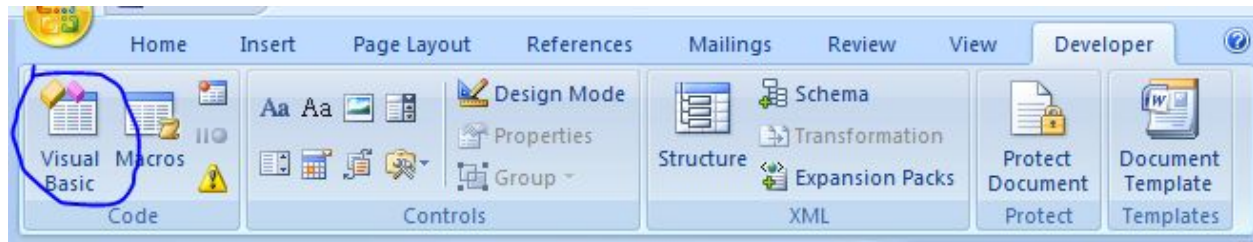


Ah, it is just asking you to enable editing... By enabling the editing you will give it permissions to enable the macros as well.

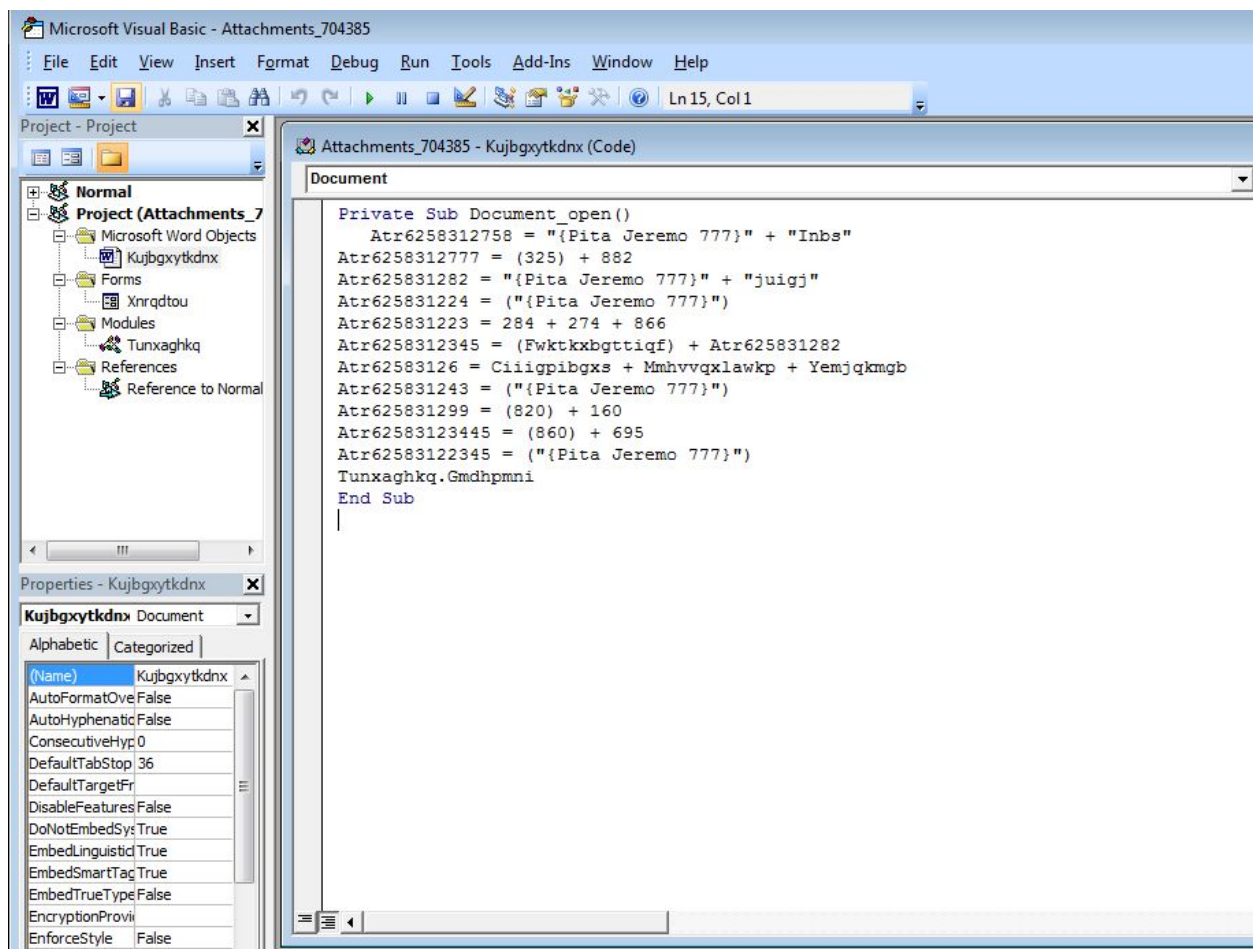
Now, what are macros? : Macros are the piece of program code that enables us to make some daily tasks automate, which can also be used for bad purposes as well... I'll demonstrate it in a minute. ;)

Let's see what macro looks like.

Just follow the below screenshot to reach macros
Click on the Developer Tab > Visual Basic.



After clicking the Visual Basic, You will see a screen like below



This is what macro looks like...

But you must be thinking this looks nothing malicious... What is wrong with this??

Let me run this doc file. And will show you what it does...

****Note:** Just keep up your network monitoring tool if you are trying with me. (i am using fiddler)**

Ah, after executing the document in the safe environment and monitoring its activity, what I saw it is making a connection with 5 URL's

****Note:** I just kept my network disabled so nothing malicious can be download**

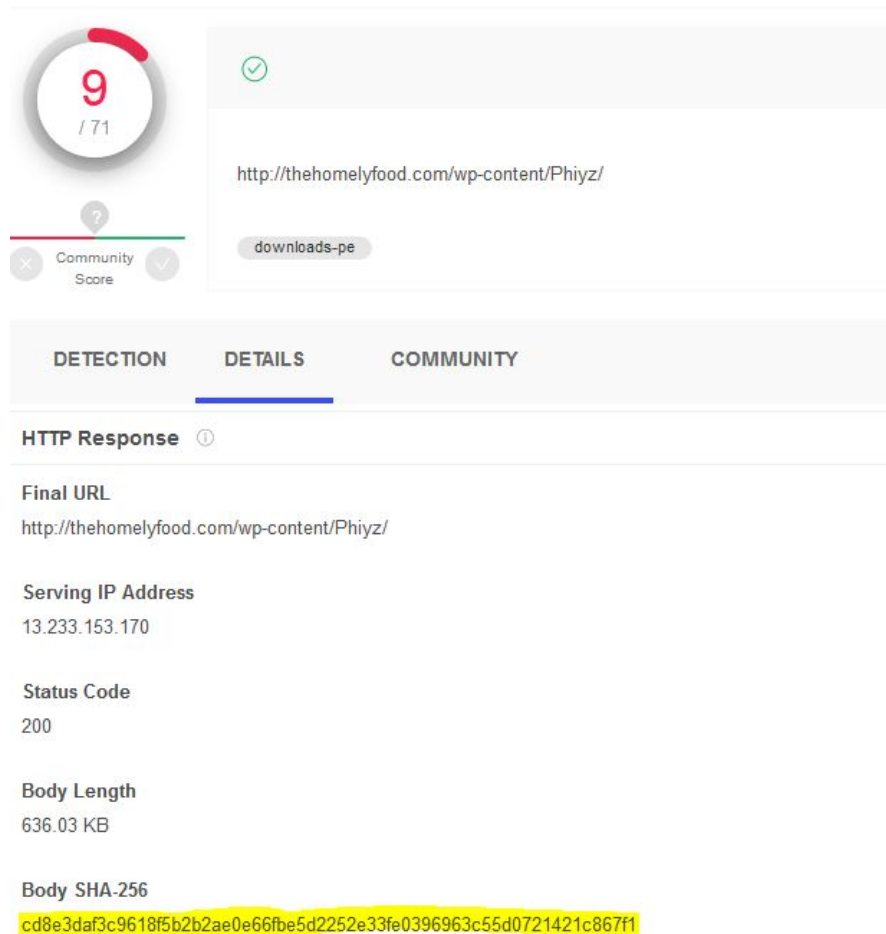
⚠️ 20	502	HTTP	thehomelyfood.com	/wp-content/Phiyz/
🔒 21	502	HTTP	Tunnel to	cfped-duca.com:443
🔒 22	502	HTTP	Tunnel to	bookdigger.azurewebsites...
⚠️ 23	502	HTTP	wasap.lse.org.ro	/wp-admin/1Dz89/
⚠️ 24	502	HTTP	furiousfox.in	/wp-content/aR/

And it is trying to drop its payload in the "c:\user\PCName" folder

But why make connections? Let's check these URLs in the virustotal.com

DETECTION		DETAILS		COMMUNITY	
CRDF	⚠️ Malicious	CyRadar	⚠️ Malicious		
Dr.Web	⚠️ Malicious	ESET	⚠️ Malware		
Fortinet	⚠️ Malware	Kaspersky	⚠️ Malware		
Netcraft	⚠️ Malicious	Sophos AV	⚠️ Malicious		
ZeroCERT	⚠️ Malware	ADMINUSLabs	✅ Clean		
AegisLab WebGuard	✅ Clean	AlienVault	✅ Clean		

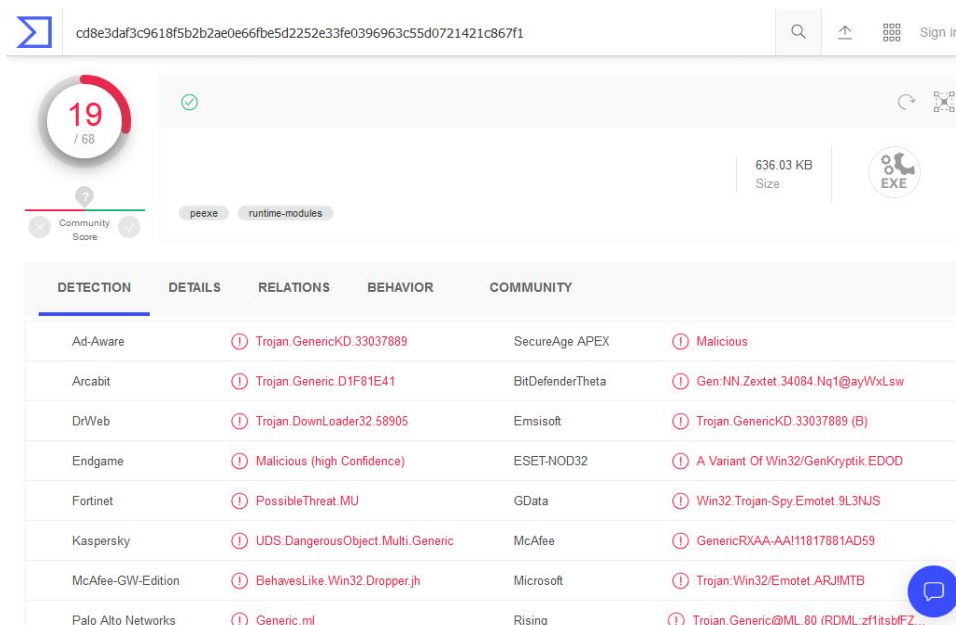
VirusTotal says the URI is malicious, but why? What is it trying to do? Let's check...



The image shows the VirusShare interface for a file. At the top, there's a circular progress indicator with the number 9 and a green checkmark. Below it, the URL `http://thehomelyfood.com/wp-content/Phiyz/` is displayed. A button labeled "downloads-pe" is visible. The interface has tabs for "DETECTION", "DETAILS", and "COMMUNITY", with "DETAILS" currently selected. Under the "DETAILS" tab, the "HTTP Response" section is expanded, showing the following information:

- Final URL**: `http://thehomelyfood.com/wp-content/Phiyz/`
- Serving IP Address**: 13.233.153.170
- Status Code**: 200
- Body Length**: 636.03 KB
- Body SHA-256**: `cd8e3daf3c9618f5b2b2ae0e66f5d2252e33fe0396963c55d0721421c867f1`

By going to its details, I got the SHA-256 of the file which this URL is trying to download, Let's click on this SHA and see what it is.



The image shows the VirusShare interface for a file, displaying detection results. The URL `cd8e3daf3c9618f5b2b2ae0e66f5d2252e33fe0396963c55d0721421c867f1` is visible at the top. The file size is 636.03 KB, and the file type is EXE. The interface has tabs for "DETECTION", "DETAILS", "RELATIONS", "BEHAVIOR", and "COMMUNITY", with "DETECTION" currently selected. The detection results are as follows:

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY
Ad-Aware	① Trojan.GenericKD.33037889	SecureAge APEX	① Malicious	
Arcabit	① Trojan.Generic.D1F81E41	BitDefenderTheta	① Gen:NN.Zextet.34084.Nq1@ayWxLsw	
DrWeb	① Trojan.DownLoader32.58905	Emsisoft	① Trojan.GenericKD.33037889 (B)	
Endgame	① Malicious (high Confidence)	ESET-NOD32	① A Variant Of Win32/GenKryptik.EDOD	
Fortinet	① PossibleThreat.MU	GData	① Win32.Trojan-Spy.Emotet.9L3NJS	
Kaspersky	① UDS.DangerousObject.Multi.Generic	McAfee	① GenericRXAA-AAI11817881AD59	
McAfee-GW-Edition	① BehavesLike.Win32.Dropper.jh	Microsoft	① Trojan.Win32/Emotet.ARJIMTB	
Palo Alto Networks	① Generic.ml	Rising	① Trojan.Generic@ML.80 (RDML:zf1jsbfZ...	

<https://www.virustotal.com/gui/file/cd8e3daf3c9618f5b2b2ae0e66fbe5d2252e33fe0396963c55d0721421c867f1/detection>

Oh look, It is getting detected by 19 AV vendors.

Now it is confirmed that Email which came to us pretending to be an invoice attached is actually a Malware Spam.

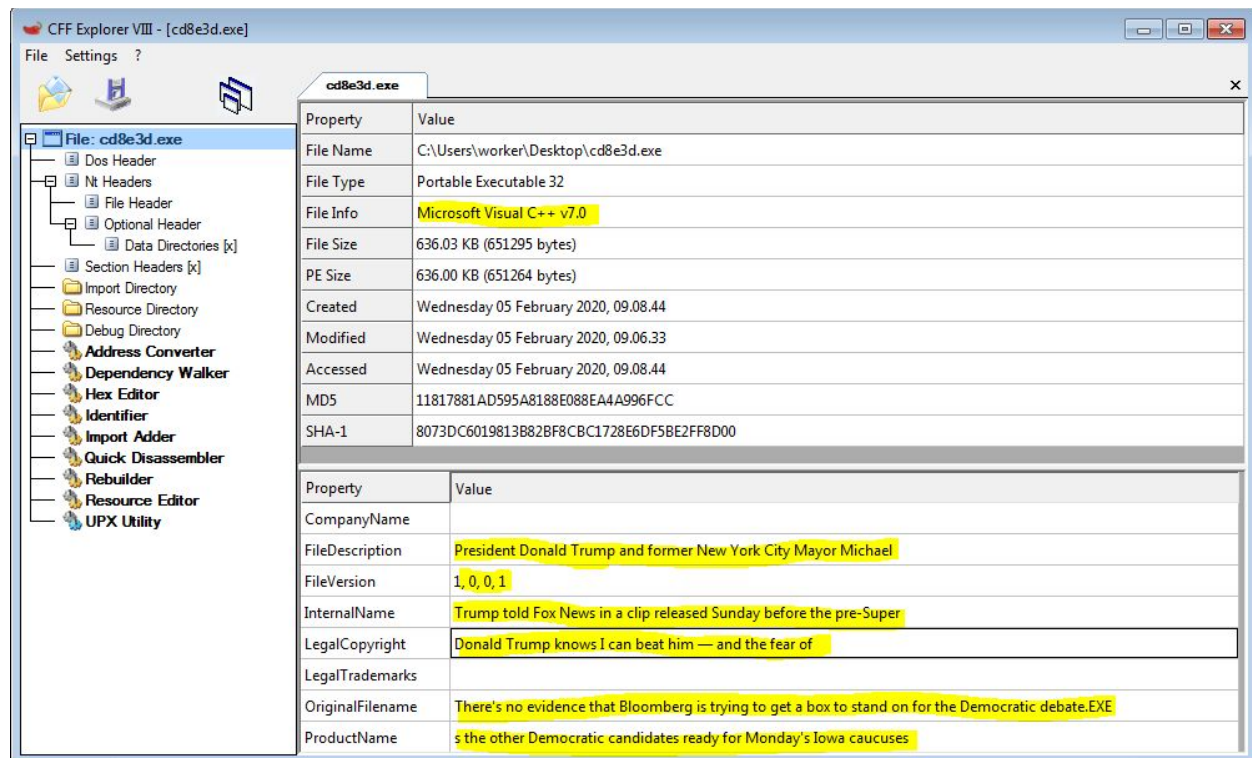
Now let's Dig more deeper into the File Downloaded by the Doc file :

SHA-256 : cd8e3daf3c9618f5b2b2ae0e66fbe5d2252e33fe0396963c55d0721421c867f1

Payload Analysis:

cd8e3daf3c9618f5b2b2ae0e66fbe5d2252e33fe0396963c55d0721421c867f1

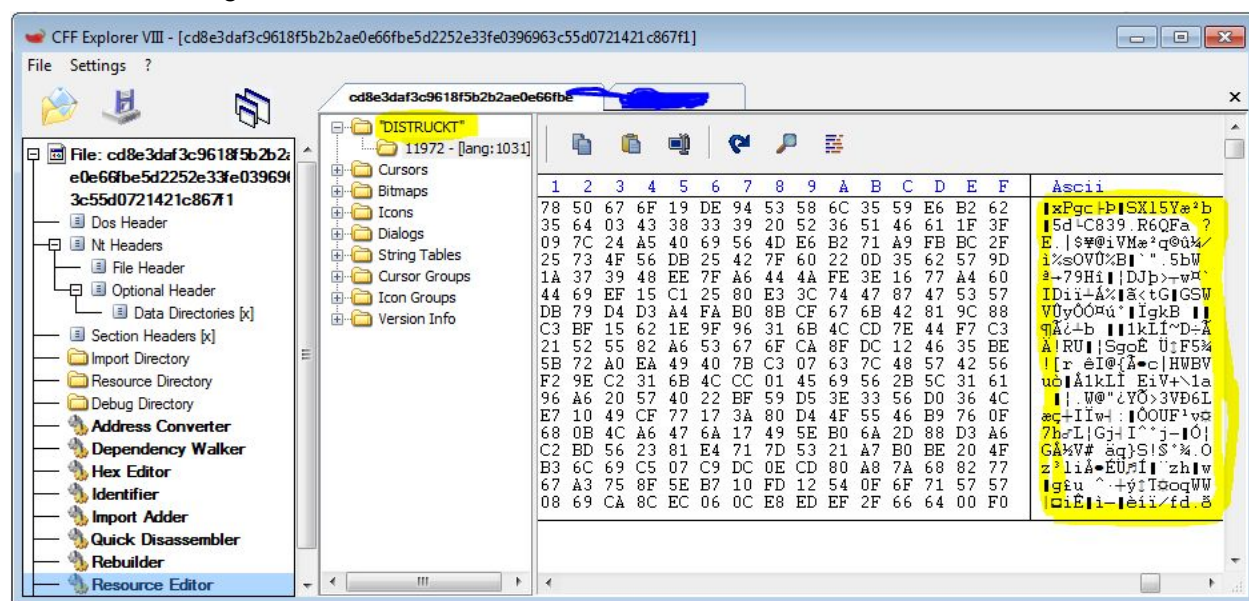
Let's open the file in CFF explorer and see what details we can gather...



Look at all the highlighted areas in the above screen-shot.

- From file info, we can say it's not packed.
- Doesn't even look like legit file description, some bogus values are entered in there related to Trump.. duh...

I found something unusual in the Resource editor as well. Check the Screenshot below.



Looking at the highlighted areas, weird “DISTRUKT” Folder in the resource section, and having too much data, the data it looks encrypted as it is so dense.

Maybe this data creates some meaningful stuff after decryption.. Will check this later. (Keep this point in mind)

Let's check strings, why? Because string analysis can tell us a lot about what the program is capable of, and see what I got... :P

```
87 DISTRUKT
88 KERNEL32.DLL
89 Fuck Sophos
90 Invalid DateTimeSpan
91 AfxOldWndProc423
```

LOL, it looks like a targeted attack to the SOPHOS, Lets see what else we have..

```
501 GetEnvironmentVariableA
502 GetEnvironmentVariableW
503 GetStringTypeExA
504 GetStringTypeExW
505 GetProcAddress
506 GetModuleHandleA

597 VirtualProtect
598 VirtualAlloc
599 GetSystemInfo

677 SetCursor
678 GetKeyState

730 GetWindowTextA

742 SetWindowsHookExA
```

It does have many more suspicious API calls.

Maybe if you're thinking how to extract strings from this file then follow below process:

Download Strings v2.53 from here: <https://docs.microsoft.com/en-us/sysinternals/downloads/strings>

Use this Command: `strings -n 8 YourFileName.exe`

Let's start debugging in the OLLyDbg

I have loaded the File in OLLyDbg. I am on the Entry point of the file.

0041F9E2	\$ 6A 60	PUSH 60	
0041F9E4	- 68 A8C74600	PUSH cd8e3daf.0046C7A8	
0041F9E9	- E8 D60A0000	CALL cd8e3daf.004204C4	
0041F9EE	- BF 94000000	MOV EDI,94	
0041F9F3	- 8BC7	MOV EAX,EDI	
0041F9F5	- E8 C6FEFFFF	CALL cd8e3daf.0041F8C0	
0041F9FA	- 8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0041F9FD	- 8BF4	MOV ESI,ESP	
0041F9FF	- 893E	MOV DWORD PTR DS:[ESI],EDI	
0041FA01	- 56	PUSH ESI	
0041FA02	- FF15 6C344600	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	pVersionInformation
0041FA08	- 8B4E 10	MOV ECX,DWORD PTR DS:[ESI+10]	GetVersionExA
0041FA0B	- 890D C42C4800	MOV DWORD PTR DS:[482CC4],ECX	
0041FA11	- 8B46 04	MOV EAX,DWORD PTR DS:[ESI+4]	
0041FA14	- A3 D02C4800	MOV DWORD PTR DS:[482CD0],EAX	
0041FA19	- 8B56 08	MOV EDX,DWORD PTR DS:[ESI+8]	
0041FA1C	- 8915 D42C4800	MOV DWORD PTR DS:[482CD4],EDX	
0041FA22	- 8B76 0C	MOV ESI,DWORD PTR DS:[ESI+C]	
0041FA25	- 81E6 FF7F0000	AND ESI,7FFF	
0041FA2B	- 8935 C82C4800	MOV DWORD PTR DS:[482CC8],ESI	
0041FA31	- 83F9 02	CMPL ECX,2	
0041FA34	- 74 0C	JE SHORT cd8e3daf.0041FA42	
0041FA36	- 81CE 00800000	OR ESI,8000	
0041FA3C	- 8935 C82C4800	MOV DWORD PTR DS:[482CC8],ESI	
0041FA42	> C1E0 08	SHL EAX,8	
0041FA45	- 03C2	ADD EAX,EDX	
0041FA47	- A3 CC2C4800	MOV DWORD PTR DS:[482CCC],EAX	
0041FA4C	- 33F6	XOR ESI,ESI	
0041FA4E	- 56	PUSH ESI	
0041FA4F	- 8B3D 1C344600	MOV EDI,DWORD PTR DS:[<&KERNEL32.GetModu	pModule => NULL
0041FA55	- FFD7	CALL EDI	kernel32.GetModuleHandleA
0041FA57	- 66:8138 4D5A	CMPL WORD PTR DS:[EAX],5A4D	GetModuleHandleA
0041FA5C	- 75 1F	JNZ SHORT cd8e3daf.0041FA7D	
0041FA5E	- 8B48 3C	MOV ECX,DWORD PTR DS:[EAX+3C]	
0041FA61	- 03C8	ADD ECX,EAX	
0041FA63	- 8139 50450000	CMPL DWORD PTR DS:[ECX],4550	

Before stepping over, Let's collaborate what information we have gathered above.

- We found a weird-looking encrypted file in the Resource section "DISTRUKT".
- It is using VirtualAlloc and VirtualProtect.
- Having some spy related modules like SetWindowHook ,GetWindowText, GetKeyState, GetSystemInfo.

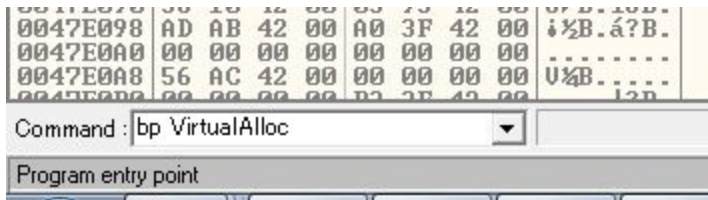
let's set some precautionary **breakpoints**, These breakpoints are totally dependent on our above analysis.

The BreakPoint on VirtualAlloc:

- VirtualAlloc is helpful because it returns newly created regions in memory in register EAX.
- And I think that our malware is trying to allocate memory regions to encrypted files inside .rsrc "DISTRUKT" and then it will decrypt the file.
- If we use a breakpoint on that, it will stop while allocating the memory and we can further analyze that allocated memory.

Note * VirtualAlloc simply allocates memory not populate it.

Setting breakpoint on VirtualAlloc:

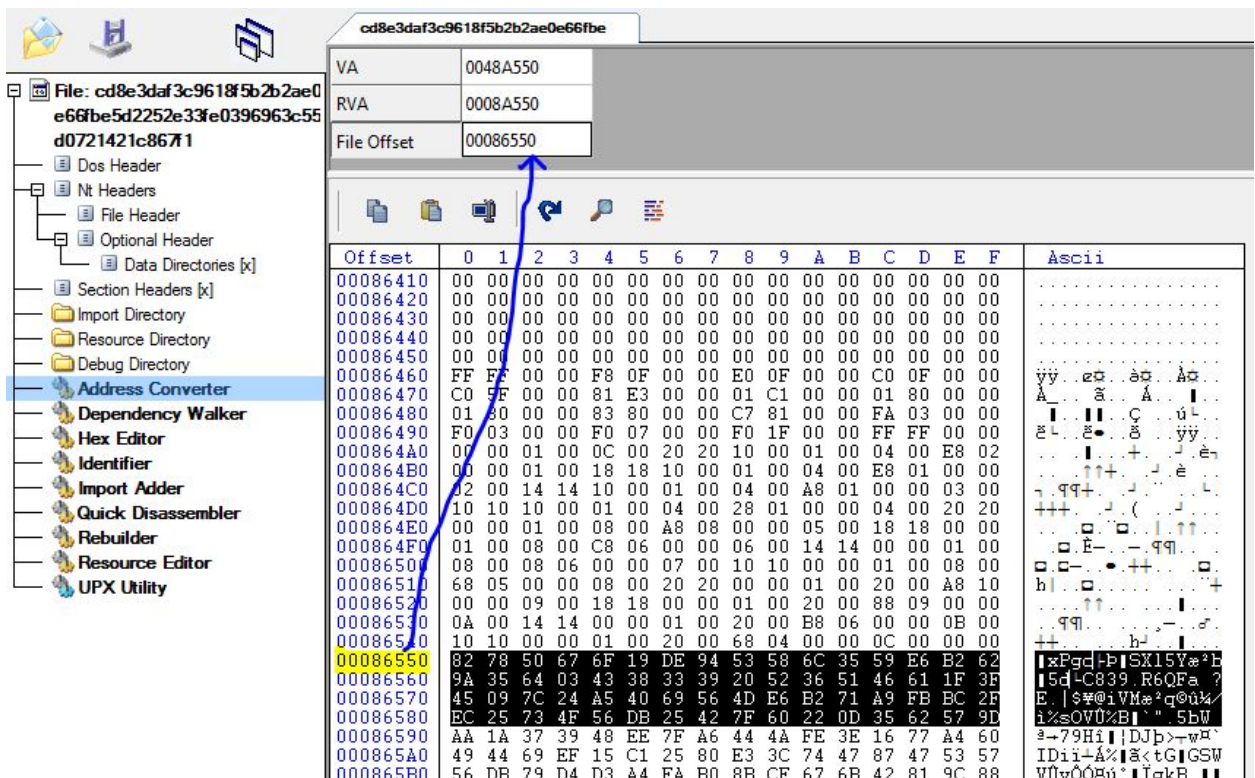


Press Enter after typing "Bp VirtualAlloc" into the command box.

The breakpoint on .rsrc memory region:

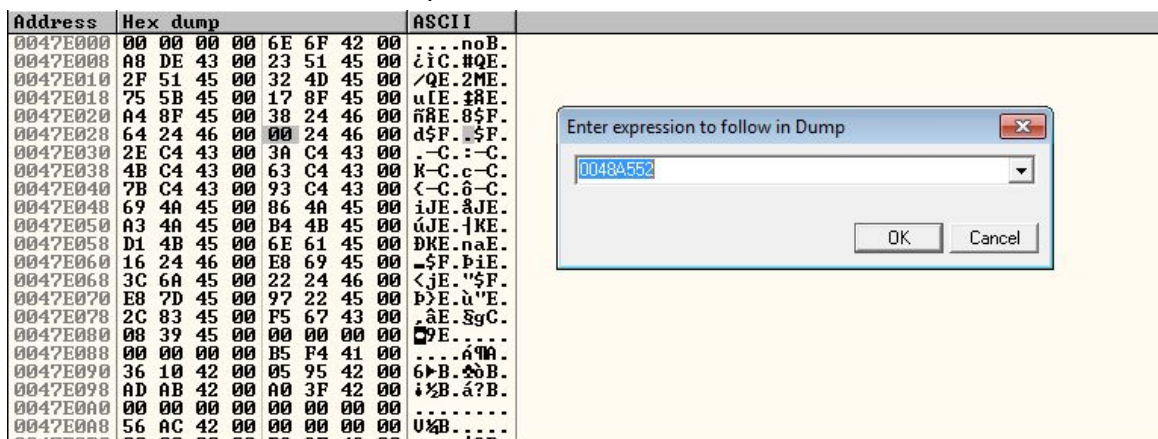
- We are going to set Hardware on access breakpoint on the memory location which I told you to **Keep point in mind**
- Because if it is going to allocate new memory for data in .rsrc, then it will surely access the .rsrc section to copy on the new location, at that point our Hardware on Access will give a pause to the program for analysis.

For setting the hardware breakpoint we have to first find the memory location address of .rsrc, Follow the Screenshot for that.



- Just take the offset of the the memory location which we looks like encrypted data in.rsrc and put it into the above converter, it will give us the VA and RVA of the file.

- Now we have VA(virtual address) of the address "0048A550" no. we have to go back to Olly in which our program already loaded.
- Press the shortcut CTRL+G and pop will appear like below, enter the memory location "0048A550" in that, and press OK

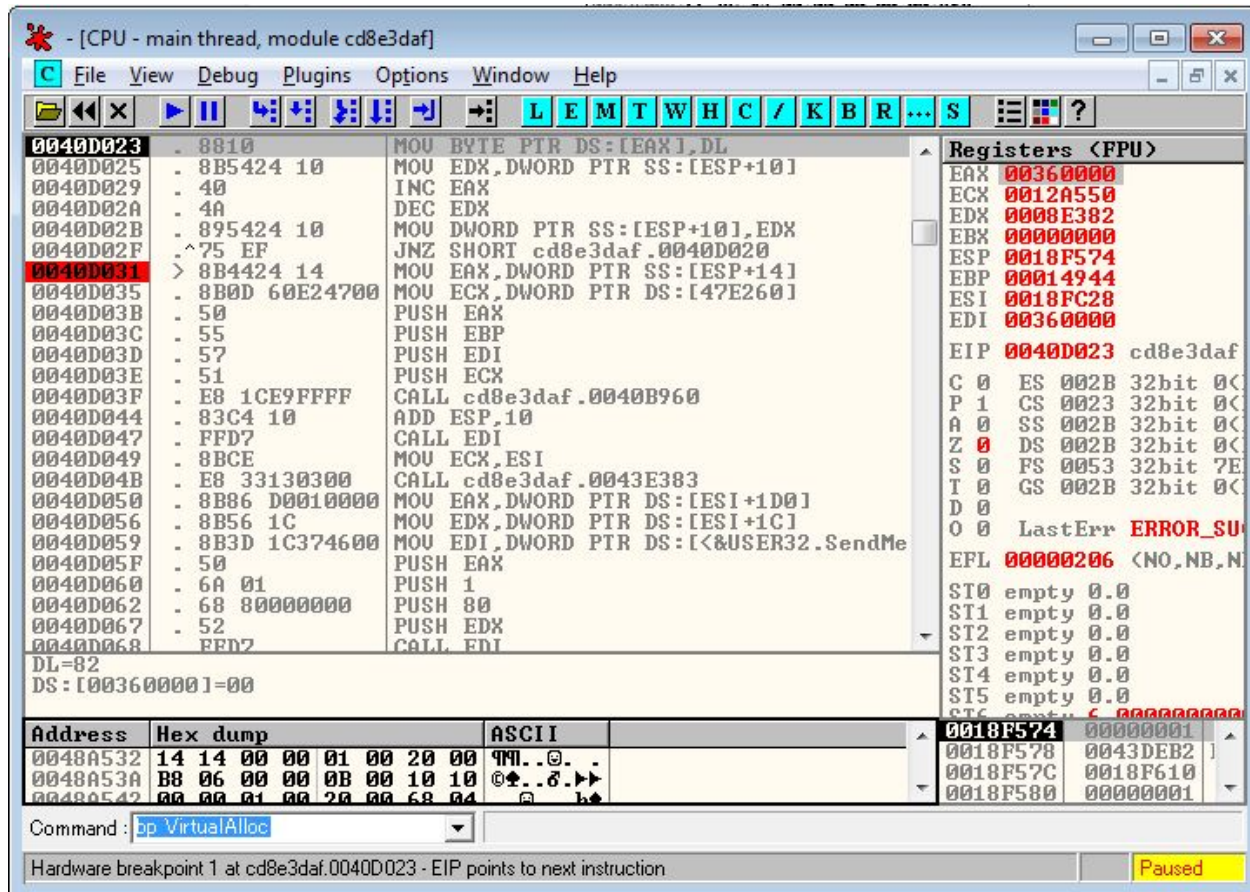


- Now you reached to the memory address "0048A550" and select some area, and now set a Hardware On-access Breakpoint by Right click > Breakpoint > Hardware, On Access > Dword

Address	Hex dump	ASCII
0048A532	14 14 00 00 01 00 20 00	...noB.
0048A53A	B8 06 00 00 0B 00 10 10	...C.#QE.
0048A542	00 00 01 00 20 00 68 04	/QE.2ME.
0048A54A	00 00 0C 00 00 00 82 78	uIE.18E.
0048A552	50 67 6F 19 DE 94 53 58	8E.8\$F.
0048A55A	6C 35 59 E6 B2 62 9A 35	d\$F. \$F.
0048A562	64 03 43 38 33 39 20 52	..C.-C.
0048A56A	36 51 46 61 1F 3F 45 09	K-C.c-C.
0048A572	7C 24 A5 40 69 56 4D E6	<-C.0-C.
0048A57A	B2 71 A9 FB BC 2F EC 25	iJE.8JE.
0048A582	73 4F 56 DB 25 42 7F 60	dJE.1KE.
0048A58A	22 0D 35 62 57 9D AA 1A	DKE.naE.
0048A592	37 39 48 EE 7F A6 44 4A	..\$F.biE.
0048A59A	FE 3E 16 77 A4 60 49 44	<JE."\$F.
0048A5A2	69 EF 15 C1 25 80 E3 3C	p>E.ù'E.
0048A5AA	74 47 87 47 53 57 56 DB	..âE.SgC.
0048A5B2	79 D4 D3 A4 FA B0 8B CF	9E.....
0048A5C2	67 6B 42 81 9C 88 14 C3â9A.

Now we set the both of the breakpoints, now we will cross our fingers and press F9.

Ah, small success, we reach the hardware breakpoint:



We got the allocated memory location in EAX "00360000", let's follow that location in the dump.

Address	Hex dump	ASCII
00360000	00 00 00 00 00 00 00 00
00360008	00 00 00 00 00 00 00 00
00360010	00 00 00 00 00 00 00 00
00360018	00 00 00 00 00 00 00 00
00360020	00 00 00 00 00 00 00 00
00360028	00 00 00 00 00 00 00 00
00360030	00 00 00 00 00 00 00 00
00360038	00 00 00 00 00 00 00 00
00360040	00 00 00 00 00 00 00 00
00360048	00 00 00 00 00 00 00 00
00360050	00 00 00 00 00 00 00 00
00360058	00 00 00 00 00 00 00 00
00360060	00 00 00 00 00 00 00 00
00360068	00 00 00 00 00 00 00 00
00360070	00 00 00 00 00 00 00 00
00360078	00 00 00 00 00 00 00 00

See the allocated memory is empty. Because encrypted data in .rsrc did not start copying yet. Keep Pressing F8 you'll see data getting copied to the allocated memory.

After pressing F8 a couple of times, we see data getting copied, let's set breakpoint outside the loop and press F9 to get the full dump.

The screenshot shows a debugger window with the following components:

- Assembly View:** Displays assembly instructions with their addresses and hex values. The instruction at address 0040D02B is highlighted.
- Registers (FPU):** Shows the state of various registers. EAX is 00360011, ECX is 0012A550, EDX is 00014933 (highlighted), and EBP is 00014944.
- Memory Dump:** A table showing hex dump and ASCII values for memory addresses starting from 00360000.
- Command Line:** Shows the command 'bp VirtualAlloc'.
- Status Bar:** Indicates the program is 'Paused'.

After pressing F9 it gave me whole data transferred.

Please mind the Address changes in next screenshot because I have restarted the program:

00360000 >to> 002F0000

Address	Hex dump	ASCII
002F0000	82 78 50 67 6F 19 DE 94	éxPgo4i8
002F0001	53 58 6C 35 59 E6 B2 62	SX15Y
002F0002	9A 00 00 00 00 00 00 00	U.....
002F0003	00 00 00 00 00 00 00 00
002F0004	00 00 00 00 00 00 00 00
002F0005	00 00 00 00 00 00 00 00
002F0006	00 00 00 00 00 00 00 00
002F0007	00 00 00 00 00 00 00 00
002F0008	00 00 00 00 00 00 00 00
002F0009	00 00 00 00 00 00 00 00
002F000A	00 00 00 00 00 00 00 00
002F000B	00 00 00 00 00 00 00 00
002F000C	00 00 00 00 00 00 00 00
002F000D	00 00 00 00 00 00 00 00
002F000E	00 00 00 00 00 00 00 00
002F000F	00 00 00 00 00 00 00 00
002F0010	00 00 00 00 00 00 00 00
002F0011	00 00 00 00 00 00 00 00
002F0012	00 00 00 00 00 00 00 00
002F0013	00 00 00 00 00 00 00 00
002F0014	00 00 00 00 00 00 00 00
002F0015	00 00 00 00 00 00 00 00
002F0016	00 00 00 00 00 00 00 00
002F0017	00 00 00 00 00 00 00 00
002F0018	00 00 00 00 00 00 00 00
002F0019	00 00 00 00 00 00 00 00
002F001A	00 00 00 00 00 00 00 00
002F001B	00 00 00 00 00 00 00 00
002F001C	00 00 00 00 00 00 00 00
002F001D	00 00 00 00 00 00 00 00
002F001E	00 00 00 00 00 00 00 00

By: Sachin Verma

contact.verma7@gmail.com

Linkedin:sachin-verma-48715a9a/

Now Press F8 until you reach the call.

```
0040D02H . 4H DEC EDI
0040D02B . 895424 10 MOV DWORD PTR SS:[ESP+10],EDI
0040D02F . ^75 EF JNZ SHORT cd8e3daf.0040D020
0040D031 > 8B4424 14 MOV EAX,DWORD PTR SS:[ESP+14]
0040D035 . 8B0D 60E24700 MOV ECX,DWORD PTR DS:[47E260]
0040D03B . 50 PUSH EAX
0040D03C . 55 PUSH EBP
0040D03D . 57 PUSH EBI
0040D03E . 51 PUSH ECX
0040D03F . E8 1CE9FFFF CALL cd8e3daf.0040B960
0040D044 . 83C4 10 ADD ESP,10
0040D047 . FFD7 CALL EDI
0040D049 . 8BCE MOV ECX,ESI
0040D04B . E8 33130300 CALL cd8e3daf.0043E383
0040D050 . 8B86 D0010000 MOV EAX,DWORD PTR DS:[ESI+1D0]
```

And press F8 on this call too. Just keep eye on HEX dump, to see data getting decrypted.

Address	Hex dump	ASCII
002F0000	E8 00 00 00 00 58 89 C3 05 3A 05 00 00 81 C3 3A	p...Xè ±:±..ü ±
002F0010	F9 00 00 68 01 00 00 00 68 05 00 00 00 53 68 45	..h0...h±...ShE
002F0020	77 62 30 50 E8 04 00 00 00 83 C4 14 C3 83 EC 48	wh0Pb±...â-¶ âúH
002F0030	83 64 24 18 00 B9 4C 77 26 07 53 55 56 57 33 F6	âd\$†.¶ Lw&±SUUW3÷
002F0040	E8 22 04 00 00 B9 49 F7 02 78 89 44 24 1C E8 14	p"±...¶ I 0xèD\$-p¶
002F0050	04 00 00 B9 58 A4 53 E5 89 44 24 20 E8 06 04 00	±...¶ XñS0èD\$ b±±
002F0060	00 B9 10 E1 8A C3 8B E8 E8 FA 03 00 00 B9 AF B1	..¶ Bè ibp±...¶ b>¶
002F0070	5C 94 89 44 24 2C E8 EC 03 00 00 B9 33 00 9E 95	\0èD\$,bú...¶ 3.x0
002F0080	89 44 24 30 E8 DE 03 00 00 8B D8 8B 44 24 5C 8B	èD\$0Bi±...iYiD\$\\i
002F0090	78 3C 03 F8 89 7C 24 10 81 3F 50 45 00 00 74 07	x<±0è ±b±u?PE..t±
002F00A0	33 C0 E9 B8 03 00 00 B8 4C 01 00 00 66 39 47 04	3 400±...0L0...f9G±
002F00B0	75 EE F6 47 38 01 75 E8 0F B7 57 06 0F B7 47 14	u÷G80uB±±W±±±G¶
002F00C0	85 D2 74 22 8D 4F 24 03 C8 83 79 04 00 8B 01 75	âEt"i0\$±Lâg±.i0u
002F00D0	05 03 47 38 EB 03 03 41 04 3B C6 0F 47 F0 83 C1	±G80±±±±;â±G-â±
002F00E0	28 83 EA 01 75 E3 8D 44 24 34 50 FF D3 8B 44 24	<â00u0iD\$4P EïD\$
002F00F0	38 8B 5F 50 8D 50 FF 8D 48 FF F7 D2 48 03 CE 03	8i PìP iH ÈH±±±
002F0100	C3 23 CA 23 C2 3B C1 75 97 6A 04 68 00 30 00 00	H±±T;±uùj±h.0..
002F0110	53 6A 00 FF D5 8B 77 54 8B D8 8B 44 24 5C 33 C9	Sj. ±iwTiiYiD\$\\3¶
002F0120	89 44 24 14 8B D3 33 C0 89 5C 24 18 40 89 44 24	èD\$¶iE3 Lè\\\$†èèD\$
002F0130	24 85 F6 74 37 8B 6C 24 6C 8B 5C 24 14 23 E8 4E	\$â÷t7i1\$1i\\±¶B¶N
002F0140	85 ED 74 19 8B C7 2B 44 24 5C 3B C8 73 0F 83 F9	âYt±iâ±D\$\\;Ls±â±
002F0150	3C 72 05 83 F9 3E 76 05 C6 02 00 EB 04 8A 03 88	<r±â±>u±±0.ù±è±è
002F0160	02 41 43 42 85 F6 75 D7 8B 5C 24 18 0F B7 47 06	0ACBâ÷uïi\\\$†±±G±

But this decrypted data still looks of no use. :(

Don't worry, scroll a little down and you'll find a hidden MZ header.

Address	Hex dump	ASCII
002F04A0	C1 EB 10 33 FF 85 DB 74 1F 8B 6C 24 14 8A 04 2F	±ù±3 â±t±vìl\$¶è±/
002F04B0	C1 C9 0D 3C 61 0F BE C0 7C 03 83 C1 E0 03 C8 47	±¶.±a±±!±â±0±L±G
002F04C0	3B FB 72 E9 8B 6C 24 10 8B 44 2A 20 33 DB 8B 7C	±rúvìl\$±iD± 3¶i±
002F04D0	2A 18 03 C2 89 7C 24 14 85 FF 74 31 8B 28 33 FF	±†±tèi\$¶â t±i±C3
002F04E0	03 EA 83 C0 04 89 44 24 1C 0F BE 45 00 C1 CF 0D	±0â±L±èD\$±±±E.±x.
002F04F0	03 F8 45 80 7D FF 00 75 F0 8D 04 0F 3B 44 24 18	±0E±> .u-ì±±;D\$†
002F0500	74 20 8B 44 24 1C 43 3B 5C 24 14 72 CF 8B 56 18	t iD\$-C;\\\$¶h±xìU†
002F0510	85 D2 0F 85 6B FF FF FF 33 C0 5F 5E 5D 5B 83 C4	âè±±âk 3 L ^1iâ±
002F0520	10 C3 8B 74 24 10 8B 44 16 24 8D 04 58 0F B7 0C	± it\$±iD±\$ì±X±±.
002F0530	10 8B 44 16 1C 8D 04 88 8B 04 10 03 C2 EB DB 4D	±iD±-ì±èi±±±±tù±¶
002F0540	5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8	Zè.±...±±...±0
002F0550	00 00 00 00 00 00 00 40 00 00 00 00 00 00 000.....
002F0560	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002F0570	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00i...¶
002F0580	1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 69	¶ ¶.±.±!±0L=±Thi
002F0590	73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F 74	s program cannot
002F05A0	20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 6D	be run in DOS m
002F05B0	6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 65	ode.....\$.....e
002F05C0	CB 6B F7 21 AA 05 A4 21 AA 05 A4 21 AA 05 A4 FC	±k ±±±±!±±±!±±±±³
002F05D0	55 CE A4 22 AA 05 A4 21 AA 04 A4 20 AA 05 A4 2C	U±i±"±±±!±±±± ±±±.
002F05E0	F8 DA A4 20 AA 05 A4 2C F8 E5 A4 23 AA 05 A4 5C	0±i± ±±±.00i±±±±\
002F05F0	D3 E0 A4 05 AA 05 A4 5C D3 EB A4 20 AA 05 A4 52	è0i±±±±±\è±±± ±±±R
002F0600	69 63 68 21 AA 05 A4 00 00 00 00 00 00 00 00 00	ich!±±±.....
002F0610	00 00 00 00 00 00 00 50 45 00 00 4C 01 05 00 1APE..L0±.±
002F0620	AD 26 5E 00 00 00 00 00 00 00 00 00 E0 02 01 0B	±8±^.....0.000
002F0630	01 0C 00 00 CF 00 00 00 54 00 00 00 00 00 00 9D	0...±!...T.....0

Linkedin:[sachin-verma-48715a9a/](https://www.linkedin.com/in/sachin-verma-48715a9a/)

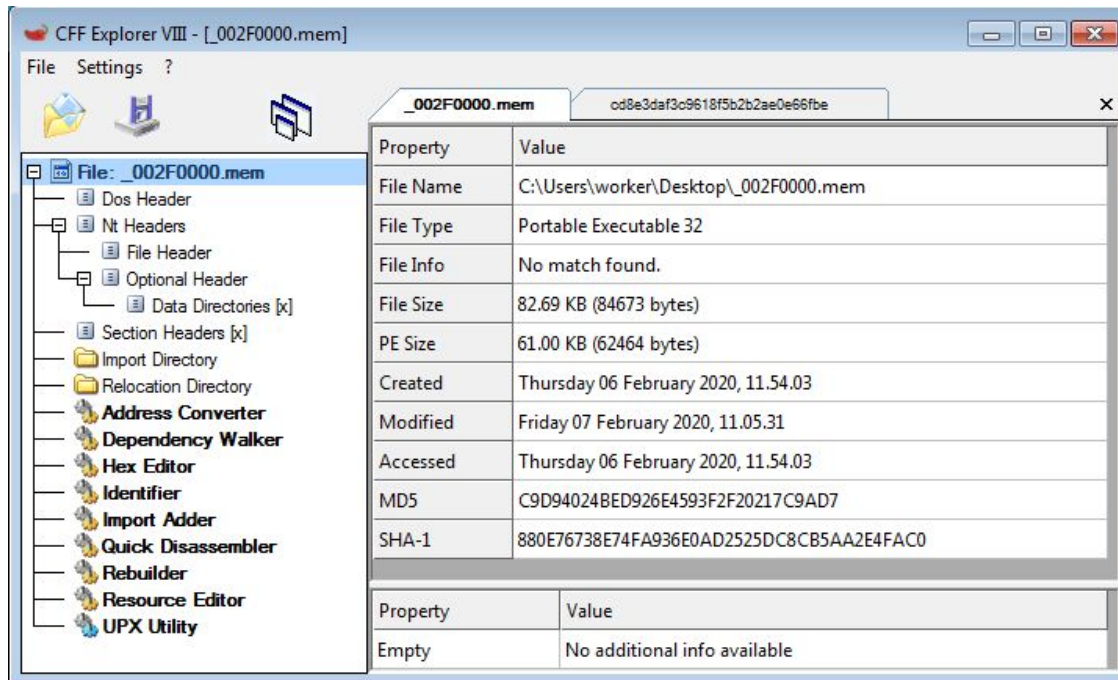
```

1  %NULNULNULNULX%AENO: ENONULNULÄ:üNULNULhSOHNULNULNULhENONULNULNULShEwb0PèEOTNULNUL
NULfÄDC4ÄfihfdS CANNULLw&BEP3SUVW3öè"BOTNULNULI-STXxtD$ES=DC4EOTNULNULX&S&tD$
èACKEOTNULNULDLEä$Ä<èèüETXNULNUL"±"%D$,èiETXNULNUL3NULZ%&D$0èfETXNULNUL<Ø<D$<x<
ETXø%$DLEPPELNUBEB3ÄÄ,ETXNULNUL,LSOHNULNULf9GEOTuiöG8SOHuèSI-WACKSI-CDC4..Ot"%O$
ETXÈfyEOTNULSOHUNEOETXG8èETXETXAEOT;ESTIGöfÄ(fèSOHuäD$4PyÖ<D$8<PPYHy-ÖHETXiETXÄèÊ#Ä;
Äu-jEOTHNUBNUBNUBSjNUByÖ<wI<Ø<D$<3È&D$DC4<Ó$Ä&\$CAN&D$$.öt7<1$1<\$DC4#èN..itBM<Ç<D$>
ÈsSIfü<rENOfü>vENOESTXNULNULEOTSETX-STXACB..öu<\$CANSI-GACKSI-ODC4..ät8fÇ,ETXi<|S<QøH<1
ETXÜ<iüETX%&D$>..itSIACK-STXBffSiSOHuö<D$>fÄ(Äüö<|DLE<èNUBNUBNUBETXö&t$DC4,FEÈÄt;
ETXÄPyT$<nDLE<ø<ACKETXèETXÄ&D$>\}NUBNUBtö<t$
<BS..étRSyES<G<SI-É<D&x+L8DLE<D8ESEOT^<EOT&ETXÇèES,ENULfÄSTXETXÄPWYÖèENULfÄEOT<D$>fÄ
EOT&D$>\}NUBNUBu<t$DC4<F
fÄDC4&t$DC4..Äu<|<DLE<è+öfç&NULNULNULNULSI,"NULNULNUL-NULNULNULETXÖ&t$JEOt<SOH%
L$DC4ÄSI,,NUBNUBNUB2xoETXöBBSNIt&D$ t'jSTX<ØZSI-VNOf<ÄfÄèFFfæ
2  tACKfføETXUv7äySINULNULSOH,1è'f;D$SuDC4äySINULNULÄÄèDLEfSOHEOT1èSI f;Äu
3  äySINULNULfSOH,1ETXÜ..yü<|\$CAN<T$<L$DC4ETXDC1&t$JEOt<SOH&L$DC4ÄSI..wyyy<|DLESI-G
ACKSI-ODC4ÄSI,,NUBNUBNUBt$<ø<ETXèHfj|NULD$S$SI+"NULNULNULMNUL3ÖB<ÄÄGS#ÄNÄèRSfä
SOHÄèUS..ÄuCAN..öuBEIjES'jSOHèENQJEOt^jSTX..èXSIDèè,..öuETB..èuEOTjDLEèNAK..öuVT..étCAN%è
NULNULNULèDC4..èuENOj ^èBSj@..ÈXSTIE&MNUL&E
4  NULSTXNULNULÄNULNULNULEOTSIDè&D$ (P<èèVyuIETXÄPyT$<ÄSI,iüyy<D$S$FÄ(ÄSI..Ryyy<w(jNUBj
NUBjyETXöYt$<3ÄèPPSyöf|S^NUBt|fÜ|NUBtv<OxETXèACAN..ätjfyDC4NUBtd<i
<y$ETXèfd$>NULETX..ätQ<uENULNUL63öSI&ACKÄÈ
5  ETXèFE~yNUBuh9TS'tSYN<D$>fÄEOTèfÇSTX&D$>ACANrèèUSSI-ETBfüýtETB&AFSyt$hyt$èEOT<EOTCAN
ETXÄyPYX^Ä^j]fÄHäfiDLEd;ONULNULNULSUV<èFFW&L$CAN<pFEè$NULNULNULF03è<^,<6&D$DC4<B<<1
DLExt&t$DLE..itmäèDLE3y..ÜtUS<1$DC4$EOT/ÄÈ
6  <aSI#Ä|ETXfÄèETXèG;üè<1$DLE<D* 3Ü<|*CANETXÄ&,$DC4..yt1<(3yETXèfÄEOT&D$ESSTI&ENULÄI
7  ETXèèè;YNUB&EOTSI&DCANT
<D$ESC;\$DC4rI<VCAN..öSI..kyyy3Ä^j]fÄDLE&t$DLE<DSYN$EOTXSI-EEDELE<DSYN$EOT^<EOTDLE
ETXÄèÜMZNUBETXNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUB
NUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUB
NUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUBNUB
8
9  $NULNULNULNULNULNULNULèÈk-!-ENOè!-ENOè!-ENOèüUfè"-ENOè!-EOTè-ENOè,øÜè
-ENOè,øâè#-ENOè\öàèENO-ENOè\öÜè

```

Now save the file after removing it. And load the file inside CFF explorer

Aha, See we got the perfect PE file ;)



SHA - 1: 880E76738E74FA936E0AD2525DC8CB5AA2E4FAC0

Checking the behavior of an Extracted file.

Now let's dive deep into the extracted PE file..... Will continue in some days.